NTNU – Trondheim
Norwegian University of
Science and Technology

Plagiarism Detection

An Overview of Text Alignment Techniques

Erisa Perleka

Master of Science in Computer Science
Submission date: July 2013
Supervisor: Bjørn Gamback, IDI
Co-supervisor: Erwin Marsi, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

Plagiarism detection is the task of identifying documents that are derived from an original source document, without giving credit to this last one. This act is aided by the modern technology, but techniques for detecting it are also ameliorated.
In this thesis an overview of plagiarism, some detection systems and the theoretical foundations, upon which they reside are discussed.

Preface

This report is the result of the requirements of the course TDT4900- Masters degree in Computer science -with a specialization in *Intelligent Systems*.
I want to thank my supervisor, professor Björn Gambäck, for his advices, input and feedback during this semester, and nonetheless for his patience in my delays.
I want also to thank Håkon Drolsum Røkenes for all his helpful explanations, and for restarting the database every time it crashed.

Erisa Perleka
Trondheim, July 25,2013

Abbreviations

NLP - Natural Language Processing

IR – Information Retrieval

POS – Part Of Speech

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Overview

1.1 Background and Motivation

Plagiarism is an old phenomenon. It has been present since humans started to tell stories and write them down in words. It can manifest itself in different disguises, making it sometimes difficult to infer a clear judgement about the originality of some work. The question of under which systematic examination a work can be precisely doomed as plagiarised or not, doesn't have a definite answer. It is of course easy for a human reader to identify cases when the passage is a verbatim copy of another. This passage can be a whole document or even a sentence, a phrase,or a mere metaphor or other figures of speech. The act of detecting the misdeed boils down to identifying the lexical similarity between the texts. Detecting paraphrases of original works also falls under the category of simple tasks; here the problem is slightly more advanced as the content words may be exchanged with words of different lexical form but same meaning, broadly referred to as synonyms. Here the reader has to derive the judgement by calculating with the meaning of words. Or we could face the case when the plagiariser has stolen the structure of a passage or sentence, something which is up to some point or completely covered by the syntax of the sentence, so here a syntactic analysis could reveal the similarity. The more complicated case of whether a text shows clear features of influence by another, falls outside the topic of this thesis.

The development of text processing and distributing technologies has given new advantages to both the plagiarisers and the techniques for detecting plagiarism(Jones, 2009). The web provides a vast amount of information, offering to a possible plagiarist what he now may need about a specific topic. The increasingly improved search engines do provide this information in critical time and the digitalized text is easily copied over in a matter of seconds. But on the other hand, these same strengths of the technology can assist a human detector in a plagiarism-hunt. It is impossible for a human to skip through million of possible source documents in reasonable, practical time, while information retrieval systems can complete the task in short time. It is also difficult for a human to remember details of texts he has already read, and find these same details reproduced in a plagiarised text, while a plagiarism detection system is better suited to find the needle in the haystack. The qualities still lacking in plagiarism systems are those of identifying

similar meanings in acclaimed different texts, or detecting an overall similar structure.

In this thesis we focus on that part of plagiarism detection that has to do with finding similar passages between two given texts, thus skipping the step of retrieving candidate source texts on the basis of a given suspicious document. In the text alignment subtask we follow the approach of combining various similarity measures in order to produce a score on the similarity between two texts.

1.2 Goals and Research Questions

Goal 1 Find existing solutions and methods for the plagiarism detection systems in the NLP literature.

This goal is reached by researching the literature available, about NLP and more specifically automated plagiarism detection systems.

Goal 2 Enhance the performance of the system described in (Røkenes, 2012) by

- Implementing an intermediate step of paragraph retrieval and improve the weights of the edit-distance function.
- Improve the precision of paragraph detection by accounting for sentences that are part of plagiarised paragraphs but give low score of similarity.
- The system shall still scale to run and be tested on the data of the PAN[1] challenge which is to be introduced later in this paper.

In order to achieve this goal we have to acquire an insight into the workings of the system, both in the overall architecture and the in the low level of the implementation. We have, moreover, to do a survey of the theoretical background upon which the system is build.

Goal 3 Combine the analysis of the (Røkenes, 2012) with the features obtained by another plagiarism detection system that accounts for semantic similarity between text snippets.

This goals fulfilment, requires that we investigate the workings of this other system, and do the implementation that allows for an integration of these two systems.

Research question 1 Given a set of retrieved source documents, is it possible to reduce the search space and consequently the number of times the algorithm is run, by doing a further paragraph retrieval, i.e. filter out those paragraphs, in both the suspicious and source documents that seem to be most similar? In that case, how can this best be done?

Research question 2 How to improve the detection of whole paragraphs, when sentences adjacent to an already detected sentence, get a lower similarity score, even though part of the plagiarised section?

Research question 3 Can graph edit distance methods, in particular the algorithm used in the (Røkenes, 2012) approach, get combined with methods that measure semantic relatedness/similarity of texts?

[1] http:pan.webis.de

1.3 Research Method

In order to attempt an experimentation in the field we had first to review the state of the art of automatic plagiarism detection systems. This did also ask for a further review, that of background material, most of which was unknown to us. Most of the basic knowledge about NLP was derived from the Martin and Jurafsky's book on the topic (mar, 2011), which was curriculum for the TDT4275 course. The slides of the TDT13[2]course were also reviewed, when we wanted to learn more about Computational Semantics. Since this work relies primarily on a previous master thesis[3] and the prototype system that was developed in its context, much of the initial research was constrained by the themes covered in it. These themes included: *graph-based representation of natural language sentences*, *graph-edit distance algorithms* and *assignment problem*. Since this system is built on the assumption that it will be trained and tested by the datasets offered by plagiarism detection competition held in the PAN workshop at CLEF[4], we also had to review the techniques and concepts relevant to this workshop.

The papers produced by the previous PAN participants, have also been an important part of the curriculum, as PAN claims to promote a state of the art situation of the detection techniques.

When using the web, the main keywords that were used in searching were :*plagiarism*, *plagiarism detection* and *text alignment*. The subcategories that fall under these fields are listed in the Table 1.2.

Of course when doing literature research one has to pick only a portion of the articles retrieved on a given topic. When deciding which articles would be considered relevant, the criteria that we used for the selection were:

1. Does the article describe a solution to the problem topic we have? And is it on the front line of research?

2. Does the article provide commanding background material that underlies the understanding of the field? In other words, is the material was good enough to be used in the theoretical part of the thesis?

Some of the articles that were cited in a given relevant article were also exploited for gaining further insight. Sometimes this was necessary, as the understanding of a work that relies on another craves knowledge of the latter one. The relevance of an article was mostly judged by reading the abstract or introduction and/or the conclusion. When this was not enough, we had to flip through the whole thing. Most of the articles retrieved where provided from the sources given in Table 1.1.

[2]http://www.idi.ntnu.no/emner/tdt13/

[3]Røkenes,Graph-based Natural Language Processing,2012

[4]http://www.clef2013.org/

Source	Url
IEEE Xplore	http://ieeexplore.ieee.org/Xplore/guesthome.jsp
Springer Link	http://springerlink.com
CiteSeerX	http://citeseerx.ist.psu.edu

TABLE 1.1: Sources of documents with background knowledge.

Plagiarism	**Natural Language Processing**	**Plagiarism Checker**
Plagiarism	Dependency Grammar	CaPD
Automated Plagiarism Detection	Parsing	
	Dependency Parsing	
	Semantic Similarity	
	Language Models	

TABLE 1.2: Keywords used in search.

1.4 Report Structure

In this section we presented among others, the method used to explore the field. In the following chapter we shall go through the concepts and definitions that we consider necessary to comprehend the rest of the thesis. After we have understood the basic concepts and principles that are important to automatic plagiarism detection systems and natural language processing, we will present the prototype we have implemented to test the ideas described in the background material. In the end we present an evaluation of the system and close with our conclusions and proposals to further work.

Chapter 2

Theory and Background

Plagiarism detection by an application is a topic which subsumes the even more fundamental matter of how to make computers cope with natural language in general, may the task include language understanding, language generation or both (Chowdhury, 2003). The issue of natural language processing has been addressed, as an important aspect of artificial intelligence, since the early years of computer science, the most important milestone being Turing's 1950 paper "Computing Machinery and Intelligence" (A.M.Turing, 1950), where the Turing-test[1] is sketched and proposed. Using natural language (besides reasoning, having knowledge and being able to learn) is one of the criteria to be fulfilled in the test. This aim is hitherto addressed by trying to formalize natural language down to the point of making it processable by the application (Jan Hajic, 2009). In this chapter we show, among other things, some of these formalization techniques.

Our plagiarism detection goal can thus be transfigured into the goal of finding similarities between these formalized structures of language. How we define and measure the similarity is strictly dependent on the specific representation chosen. Various plagiarism techniques ask for different detection strategies. Plagiarism is a product of the human mind; it is though a highly nondeterministic event. In this paper, we confine our goal to detecting the plagiarism on document level, which means, finding which passages of a suspicious document correspond to and can be claimed equivalent to passages that are to be found in some source(original) documents. This task creates two subtasks: the first is finding the source document from which the suspicious document has stolen the passage. The second is pointing exactly to the passage plagiarised. Again, these two tasks can be broken down into other tasks, but in the bottom of both of them, lie three problems: the first, also mentioned above, that of representing natural language in computational terms.The second that of determining when two passages are enough similar for the case to be an instance of plagiarism (and this is discussable even in human terms) and the third, to develop techniques that measure these similarities.

[1] http://plato.stanford.edu/entries/turing-test/

2.1 Plagiarism

This section presents what in general is understood by plagiarism and discusses some instances of this phenomenon that are of interest to this thesis.

2.1.1 The Concept

The term *plagiarism* appears for the first time as late as in the 1st century, when it was coined by the latin poet Martial, in the 1.52 passage of his *"Epistemmes"* Seo (2009). It has since then been used to define the theft of not only language artifacts, but of any work that beares authorship and originality. We shall however confine our study to the detection of plagiarism in texts.
Plagiarism is the act of using someone else's work without giving a proper recognition of its original author. Although the concept and its connotation are familiar, it has given rise to a variety of definitions, thus making vague, the parameters that would decide if a work can be considered as plagiarised or not.This lack of clarity of the definition also contributes to the difficulty of automating plagiarism detection. Sometimes it can be unambiguously clear that a certain piece of text is a mere copy of another. But plagiarists often use techniques that alienate the new, plagiarised text from the source. Here the problem becomes more intriguing, but it still remains decidable to a human reader, when faced with both the source and the suspicious text. Here is a definition that lies most nearly to our understanding of the concept:

> "The wrong in plagiarism lies in misrepresenting that a text originated from the person claiming to be its author when that person knows very well that it was derived from another source, knows that the reader is unlikely to know this, and hopes to benefit from the reader's ignorance." (Samuelson, 1994)

The core phrase that serves our purposes is stating that a plagiarised text is "derived from another source". The task of an automatic plagiarism detector, is primarily to find out from which source the text is taken, but also argue about the nature of the specific derivation used. Showing what kind of plagiarism the text has been a victim of, is more persuasive than giving a plagiarised/not-plagiarised binary classification. Upon which rules a plagiarism checker should make its examination, is one of the oldest and still relevant research problems. So one of the goals for the automatic detection is to find suitable, quantifiable features that can be used as parameters in the detection (Clough, 2003). In the following we show some instances of plagiarism, to make concrete the tasks a detector has to accomplish.

2.1.2 Kinds of Plagiarism

To illustrate the concept and the kind of problems we are faced with, we present here some examples of plagiarising from a text, taken from the website of Georgetown University. [2] The recycled text is typed in red, the reformulated phrases or sentences are in

[2] http://gervaseprograms.georgetown.edu/honor/system/53501.html

blue:

THE ORIGINAL PASSAGE

> This book has been written against a background of both reckless optimism and reckless despair. It holds that Progress and Doom are two sides of the same medal; that both are articles of superstition, not of faith. It was written out of the conviction that it should be possible to discover the hidden mechanics by which all traditional elements of our political and spiritual world were dissolved into a conglomeration where everything seems to have lost specific value, and has become unrecognizable for human comprehension, unusable for human purpose. Hannah Arendt, The Origins of Totalitarianism (New York: Harcourt Brace Jovanovich, Inc., 1973 ed.), p.vii, Preface to the First Edition.

EXAMPLE I: Verbatim plagiarism

> This book has been written against a background of both reckless optimism and reckless despair.It holds that Progress and Doom are two sides of the same medal; that both are articles of superstition,not of faith.Interestingly enough, Arendt avoids much of the debates found in some of the less philosophical literature about totalitarianism.

EXAMPLE II: The Paraphrase

> Hannah Arendt's book, The Origins of Totalitarianism, was written in the light of both excessive hope and excessive pessimism. Her thesis is that both Advancement and Ruin are merely different sides of the same coin. Her book was produced out of a belief that one can understand the method in which the more conventional aspects of politics and philosophy were mixed together so that they lose their distinctiveness and become worthless for human uses

EXAMPLE III: The Mosaic

> The first edition of The Origins of Totalitarianism was written in 1950. Soon after the Second World War,this was a time of both reckless optimism and reckless despair. During this time, Dr. Arendt argues, the traditional elements of the political and spiritual world were dissolved into a conglomeration where everything seems to have lost specific value. In particular, the separation between the State and Society seems to have been destroyed. In this book, she seeks to disclose the hidden mechanics by which this transformation occurred.

EXAMPLE IV: The "Apt Phrase"

> Following the Second World War, scholars from a variety of disciplines began to explore the nature of "totalitarianism." One of the most pressing issues for these writers was understanding the "essence" of totalitarianism. How, for example, is a totalitarian regime different from an authoritarian regime? Although authors disagree on the precise answer to this question, a common thread running throughout most of the classic works on totalitarianism deals with the relationship between State and Society. In a totalitarian state, the traditional boundaries between State and society are dissolved into a conglomeration so that the two become indistinguishable.

To define the plagiarism methods, we follow (Clough, 2003) who distinguishes six ways of doing plagiarism. The list below is taken from (B.Martin, 1994):

1. **Word-for-word plagiarism**: direct copying of phrases or passages from a published text without quotation or acknowledgement. This is what we see in Example I and is also the easiest to detect. The algorithm could simply identify the perfect one-to-one match between the lexemes of the text.

2. **Paraphrasing plagiarism**: is the case when words or syntax are rewritten, but the source text can still be recognised. This is illustrated in Example II. The task is more complicated for an automated detector as the phrases are lexically different. Here we list the transformations:

- "*reckless despair and reckless optimism*" $\longrightarrow$ "*excessive hope and excessive pessimism*".
  Here we have both a change of lexicon and a switch of the word-order inside the phrase. In this case, the algorithm would have to know how to find out that "despair" $\rightarrow$ "pessimism" , "optimism" $\rightarrow$ hope and "reckless" $\rightarrow$ "excessive" are pairs of synonyms. In order to give a high score of phrase similarity the algorithm must also ignore the order of content words.

- "*progress and doom are two sides of the same medal* " $\longrightarrow$ "*both advancement and ruin are the merely different sides of the same coin*"
  In addition to reformulation of the phrases, here we have the insertion of "merely" and "both". Besides equating between "progress" and "advancement", "doom" and "ruin" and, "medal" and "coin", the algorithm should assign very little importance to the insertion of "both" and "merely". One could for example argue that "both" is in the second sentence a semantic equivalent of "two" in the first sentence, which is deleted. While "merely", as an adverb, has little to add to the meaning of the sentence. So the algorithm should be able to judge "merely" as redundant.
  While the last sentence of the Example II is completely reformulated, asking for the detector to disclose the similarities by detecting the similarities between the two sentence meanings. Here we encounter the problem of defining the meaning of a sentence computationally, which we shall talk about at a later point.

3. **Plagiarism of secondary sources**: When original sources are referenced or quoted, but obtained from a secondary source text without looking up the original. We will not pay much attention to this case, as it involves a chain comparison between texts. E.g, comparing text1 with text2 which is in turn compared to text3.

4. **Plagiarism of the form of a source**: The structure of an argument in a source is copied. This can be the same case as when paraphrasing by keeping the sentence structure and exchanging content words with synonyms as in Example II. Or merely copying the syntactical content of a sentence, without regard to word meaning.

5. **Plagiarism of ideas**: The reuse of an original thought from a source text without dependence on the words or form of the source. Assuming that ideas have a meaning, we could count this case as paraphrasing with different words. This can be illustrated by the last sentence of Example II.

Another method, that is not listed in (Clough,2003; Martin,1994), is that of **Back Translation**. This constitutes in translating a text, verbatim, from the original to another language and then retranslating it back to the original with the structure, words and meaning somehow changed. (Jones, 2009) argues that among other factors, this form of plagiarism is encouraged from the easiness of pursuing it, given the continuously ameliorating state of the online automatic translation applications. For example, by using Google Translate[3]. In the case below, the application obfuscates an english sentence by translating it to Italian and than back to English:

"*They have lived here all their lives, they know the land*" $\longrightarrow$ "Hanno vissuto qui tutta la vita, conoscono il territorio "$\longrightarrow$ "*They have lived here all my life, know the territory*"

In this thesis, we try to address the *Verbatim* and *Paraphrasing* cases of plagiarism since, as we shall see below, they are considered by the PAN challenge which we use as an evaluation reference.

2.1.3 Artificial plagiarism

In order to develop an intelligent plagiarism checker, one has to train it on a large amount of data. Test data is also needed when one has to measure the performance of the system. It is, however, quite difficult to get hold of big sets of real plagiarism cases, as these cases are often held secret and publishing them would ask for a permission from both the plagiariser and the author of the original Potthast (2011). The PAN (Plagiarism analysis, Author identification and Near-duplicate detection) evaluation lab proposes, after Potthast (2011) a corpus of suspicious and source documents, where the suspicious documents are created utilising the source documents by using three obfuscation strategies. Here s_{plg} denotes the plagiarised passage and s_{src} denotes the source passage. The below definitions are taken directly from Potthast (2011):

[3] http://translate.google.com

- Random Text Operations : s_{plg} is created from s_{src} by shuffling, removing, inserting, or replacing words or short phrases at random. Insertions and replacements are taken from the document d_{plg} where s_{plg} is to be inserted.
- Semantic Word Variation: s_{plg} is created from s_{src} by replacing words with their synonyms, antonyms, hyponyms, or hypernyms, chosen at random. A word is kept if none are available.
- POS-preserving Word Shuffling: The sequence of parts of speech in s_{src} is determined and s_{plg} is created by shuffling words at random while retaining the original POS sequence.

The drawback of the artificial plagiarism, is that is does not allow us to talk about the meaning of the sentence as a whole. It is also not reasonable to try capturing the meaning of a word by analysing its neighbours, as there is no intentional connection between them. We show here a fragment taken from a plagiarised document of the PAN13 data set, which has been prone of random obfuscation:

> ”table alcohol” in the sake existence, and make up about 80is added with distilled amounts of copious concentrated intoxicant make to addition give.

2.2 Formal Representations of Natural Language

In order for a computer application to use the knowledge that relies in a natural language utterance, it must have it represented in a way that is suitable for processing. Representing linguistic knowledge is a subtopic of the the problem of knowledge representation in computer science. As (Davis et al., 1993) argue, one of the properties of knowledge representation is that of being a surrogate, which means, being an incomplete representation of a real world concept. This quality (or lack of a quality) is also true for the problem of representing natural knowledge. What makes this problem even more complicated, is the fact that natural language is prone of ambiguities which cross it through all levels. These ambiguities are a source of confusion not only for a computer application, but even for humans. The list below, taken from Allen (2003), states some of the ambiguities.

- **Lexical ambiguity**: appears when a given morpheme can take on different part-of-speech roles, depending on the context. For example, ”light” can be a noun,as in ”the light is on” or an adjective, as in ”the package is light”.
- **Structural or syntactic ambiguity**: has to do with finding out in which phrase group a word belongs to. For example, ” I saw a man with a telescope”, here the telescope could ”be held” by the subject or by the object, and we could argue what the sentence actually denotes.
- **Semantic ambiguity**: many words have more than one meaning, like for example the verb ”go”, which has ten meanings listed in the dictionary.

- **Pragmatic ambiguity**: sometimes it is confusing what a phrase or sentence is hinting to. For example, ”Can you lift that rock?” could be a yes/no question or a request to lift the rock.

Another feature of language, is that of being redundant,which means that phrases or whole passages have elements that they can exist without both semantically, syntactically and grammatically. Like in the Example II above where deleting ”merely” from the sentence wouldn't harm it by any way.

2.2.1 Basic Text Processing

Expressing linguistic knowledge in a way that it can be processable by formal means, is a continuously developing trend as (Jan Hajic, 2009) argue. Below are listed the most widespread techniques of text processing that are shared the majority of systems that do some natural language processing. More explicitly, these are fundamental techniques in IR (**?**).

- POS-tagging : Part-of-speech tagging is also known as word-category disambiguation, as it assigns a corresponding part-of- speech tag to every word in a sentence. This is done based on the word's definition and on the context where it is declared. In compuational linguistic, POS-tag algorithms are divided into rule-based and stochastic. The first peform the tagging based on a set or rules, similar to the rule-based expert systems, while the second ones rely on probabilistic models.

- Lemmatisation: Is the process of determining the lemma of a word. Lemma denotes the cannonical or dictionary form of a word. For example the past tense ”fed” has as lemma ”feed”.

- Parsing: This is a more complicated task as it requires both the pos-tag preprocessing step and the lemmatisation of the text as a prerequisite for its own processing of the sentence. Parsing means to do a grammatical analysis of the sentence and this requires a definition of the language grammar. The three main categories of approaches to parsing are those of :formal grammars(**?**), head-driven phrase structure grammar(hea) and dependency parsing. A detailed representation of dependency parsing is going to be carried later on.

2.2.2 Bag of Words

The bag-of-words model is a very plain representation used in both natural language processing and in information retrieval(Wikipedia: Bag of Words). In this model, a text (such as a sentence or a document) is represented as an unordered collection of words, disregarding even the grammar of the sentence. The bag-of-words model is commonly used in methods of document classification, where the occurrence of each word is used as a feature for training a classifier. It is the simplest of the representations as of the information of a sentence or document it only preserves the words that are contained in it. For example, the sentence:

Example:

```
Some students study in the mornings.
Other students prefer the afternoon.
```

These two sentences can be first represented as a list of strings(words), to build the dictionary of distinct terms.

```
"Some"
"students"
"study"
"in"
"the"
"mornings"
"Other"
"prefer"
"afteroon"
```

The vectors here stand for the number of times that each dictionary word appears in the sentences. The first vector represents the first sentence, and the other the second sentence.

$$[1, 1, 1, 1, 1, 1, 0, 0, 0] \tag{2.1}$$

$$[0, 1, 0, 0, 1, 1, 1, 1, 1] \tag{2.2}$$

Term Weighting

In the example above, the sentence vectors contain term frequencies. Even though this approach may seem appropriate in the case of such short texts, it is not widely used when dealing with long text documents that are part of a large corpus. One of the most popular schemes in weighting term frequencies is tf*idf, the so called term frequency- inverse document frequency scheme. This method reveals the saliency of a word in a text compared with the use of this word in the whole collection of texts that the document is part of. As the name denotes, one takes the number of occurrences of a term in a document and roughly said, divides it by the number of times that this word appears in the corpus. If the word has a high frequency relative to the document and low relative to the corpus, than it is surely representative for the document. This scheme can be used for example to filter out the so called stop-words, words of common usage like ”the”, ”and” etc that don’t give much information about the semantic content of a text. For the mathematical derivations and the formula see (Baeza-Yates, 2011)

2.2.2.1 Standard Vector Space Model

The Vector Space Model (VSM), first used in the SMART information retrieval system (Wikipedia :Vector Space Model), is an algebraic model for representing text documents as vectors of terms. Both documents and queries are represented as vectors. The notation below shows the document *dj* and the query *qj*:

$$d_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j}) \tag{2.3}$$

$$q = (w_{1,q}, w_{2,q}, \ldots, w_{t,q}) \tag{2.4}$$

Here each dimension corresponds to a separate term whose value is different from zero only when the term exists in the document. Here is again the term-weighting problem encountered, which is on what basis should the value of an occurrence be set. The tf*idf scheme is used also here. In fact this model is tightly coupled to that of Bag-of-words, as it does treat the documents as such. In (Baeza-Yates, 2011) VSM is presented as the classic model of the algebraic frameworks but also as a standard information retrieval algorithm. Since the text is presented as a vector one can utilize the mathematical properties of standard vector algebra, the most releant of them in these case, being to calculate the similarity between two vectors which is done by calculating the cosine of the angle between the vectors.

The figure below shows the graphical representation of the cosine similarity for two documents in a corpus.

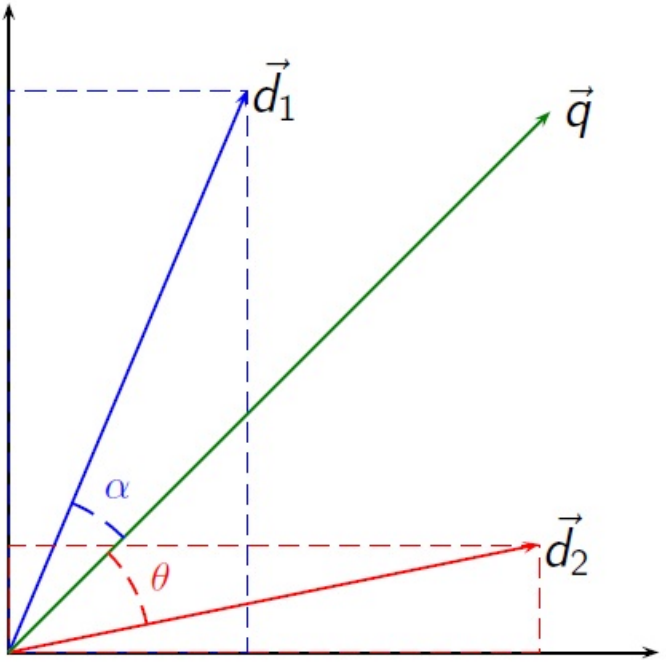

FIGURE 2.1: Document similarity measured by the cosine of the vector angles

2.2.2.2 Latent Semantic Analysis

Until now, the similarity measures that are presented dealed only with the lexical similarities between texts, which is, how may terms they share. Latent Semantic Analysis

(LSA)is a method for disclosing the meaning of words in a context, by applying some statistical computations on a text corpus (lsa). The base idea of this method is, that accounting for all the contexts in which a word occurrs (and consequently for those where it doesn't occurr) one can trace some information about what semantic similarity between two words (or sets of words) by this context-information. As a practical method for assessment of word meaning, LSA can produce measures of word-word, word-passage and passage-passage relations.

2.2.3 N-gram Models

N-gram models are probabilistic language models. By means of a probability distribution, they assign a probability to a given sequence of words:$P(x_i|x_{i-(n-1)}, \ldots, x_{i-1})$. For practical reasons, the independence assumptions are made such that a word depend only on the last n-1 words. It answers the question: what is the probability that word w1, is followed by w2 which is followed by anoter one and so over. In NLP they are widely used in statistical natural language processing, like could be the case of a POS-tagger, when this model could say what part-of-speech follows after another. So the n-gram models reduce the problem of learning the language model from the data. An n-gram of one word is called a unigram. When n equals two it is a bigram, when three, we call it a trigram. These are also the most common form of n-grams. However when discussing the issue of plagiarism detection, the n-gram models can be used to find overlapping sequences of words that exist in different text.

2.2.4 Graph-based Representation

One of the ideas behind graph-based representation of natural language is to transfer the notion of dependency between syntactical elements in sentences to the notion of directed edges in graphs. In this way, an edge pointing from a node to another would denote that the first node is dependent on the second. Graph-edit distance algorithms are used also outside graph theory. The Levenshtein edit distance algorithm is one of the oldest and most widespread ones. It is used as a similarity metric when calculating the distance between strings. It is defined as the least amount of operations needed to transform one string to the other. These operations can be deleting, inserting or editing (which is deleting and inserting). The edit distance on graphs employed by (Røkenes, 2012), employs somehow the same ideas.

2.2.5 Dependency Parsing

Dependency parsing stems from the notion of dependency grammars, which is a class of grammar formalisms. Here the syntactic structure of a sentence is described purely in terms of words and binary semantic or syntactic relations between these words. The theoretical framework of dependency grammars assumes that an essential part of the syntactical information of the sentences resides in the binary relationships between the lexical parts of the sentences, these relationships being called dependencies. To illustrate the idea, we give here the dependency graph of the sentence

1. I shot an elephant in my pajamas

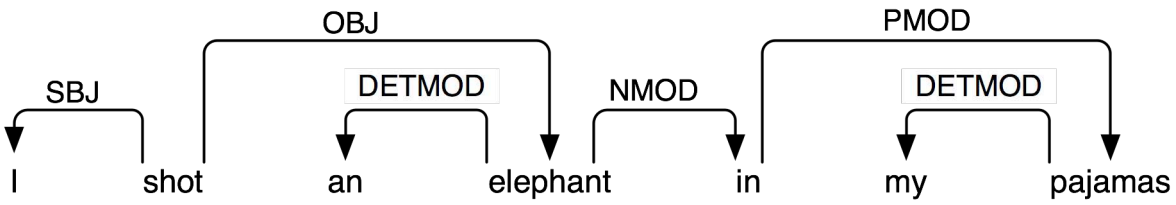

FIGURE 2.2: An example of dependencies within a sentence.

The edges are labelled with the dependency relations, the arrow pointing from the head to the dependant. What we can learn from dependency graphs is the syntactical structure of the sentence. As (Røkenes, 2012) concludes, these graphs are suitable for computing syntactic similarities between two sentences as the problem of measuring the similarity (or dissimilarity) between them can be transformed to that of finding the last cost for an assignment of changes, thus to the assignment problem. But we come nearer to this in he section on Plagiarism Detection Methods. Another convenience that follows from dependency parsing, according to (Covington, 2001) is that dependency links are close to semantic relationships needed for the next step of interpretation. This could somehow be true because

2.3 Plagiarism Detection Methods

The advantage of automatic detection is that of being able to determine the authenticity of a given text by comparing it with a large amount of other texts in a reasonable time span, something which is not feasible for a human detector. A plagiarism checker can be classified as a specialized IR system, which is built to perform the task of detecting plagiarism (Wikipedia:Plagiarism Detection). There are two major approaches to determining the originality of a document. The first is the internal detection. When one can judge it from detecting a change of style and vocabulary within a document, thus discovering inconsistencies which can be interpreted as signs of plagiarism. The other approach is that of comparing a given text, spoken of as the *suspicious text* with another one, which is regarded as the *source text* where the first one is possibly plagiarised from. This is called the external plagiarism detection. This latter form for detection uses many techniques from the area of information retrieval, as it comprises the steps of finding out relevant original documents and thereafter performing on them a detailed comparison analysis which aims to detect the concrete instance of plagiarism. The figure below shows an overview of the detection methods. In this thesis, we are concerned with the external,monolingual, local similarity assessment and somehow with the term occurrence analysis as it is employed in some systems that are described in Chapter 3.

It is inherent that plagiarism detection systems do some natural language processing, since even IR is considered as a subfield of NLP. Ideally, a system should manage to have a thorough linguistic cognition, but given the the ambiguity and the ever-changing vocabulary of natural language this is yet not feasible. The most severe challenges posed to automatic detectors come from the changes made to the original text. (Miranda Chong, 2010) list some strategies of rewriting an original text:

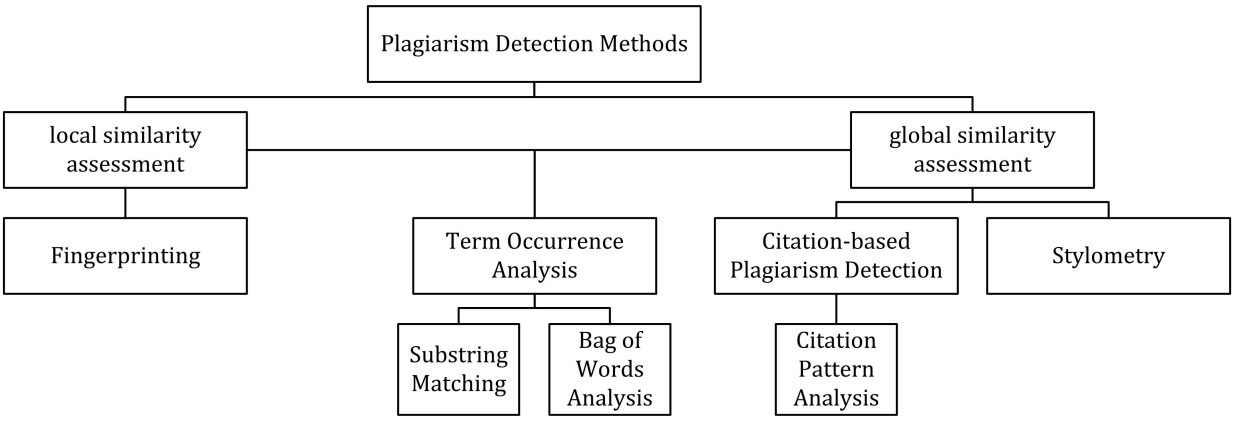

FIGURE 2.3: Classification of computer-assisted plagiarism detection methods.

- Lexical changes.
  With lexical changes is understood the substitution of content words with respective synonyms or related concepts.
- Structural changes.
  These changes may include a transformation from active to passive voice. Changing the word order without affecting the meaning of the sentence, like changing the order of the tokens that constitute an enumeration. Another technique is that of splitting long sentences or merging short ones.
- Textual Entailment.
  The authors employ this term to mean paraphrasing as discussed above in the list from (Clough, 2003)

In general the changes, or the similarities, could be grouped in two sets: semantic similarities, when the texts say quite the same thing, and structural similarities, when the texts are t

These challenges lay the foundations for the requirements that a detector should satisfy, namely: it should capture lexical changes, structural variations and detect paraphrases that are derived from some other text. One question that arises in this context is, can a detector employ strategies for all these tasks at once? Or can it only accomplish one task at a time. We shall later present the system of Kong Leilei (2012) which employs both semantic and structural similarities to detect plagiarised passages. Moreover, there are systems which employ different semantic features in order to assess the semantic similarity of two texts, like (Erwin Marsi, 2013) and (Eneko Agirre, 2012). The notion of similarity is central to the detection task. There are several comparison methodologies that can be employed to calculate similarities. The list below gives some of them:

- N-gram similarity measure
- Language Model probability measure
- Longest Common Subsequence

2.3.1 The Longest-shared-passage Problem

This problem is formulated like this: given a document that contains plagiarised text and another one from which the plagiarised passage is derived, how to detect which passage is plagiarised from which passage. The figure below is used to illustrate the case:

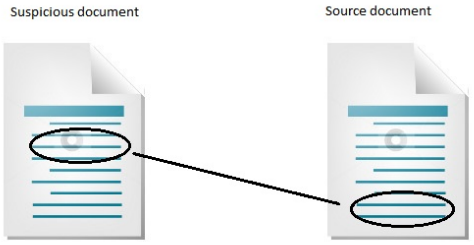

FIGURE 2.4: The Longest-shared-passage Problem

Since the scope of plagiarism can vary from that of a phrase to a whole text, it makes sense to start calculating similarities from a phrase or sentence level, and then merge these discovered sentences into a paragraph, if they are consecutive, thus discovering entire plagiarised passages in a given text. How this goal can be achieved, depends on the formal representation of language that we choose, and the respective algorithms that we can apply to that representation. This problem of detecting whole passages may these be a perfect copy of each other or a blurred one, is referred to as the longest common subsequence problem, which is discussed below.

2.3.1.1 Sequence Alignment

Sequence alignment is also cited as the longest common substring problem. This definition is somehow borrowed from the field of bioinformatics because the techniques used to solve the problem stem from there where it is used in the Genome project, in order to identify regions of repeated genetic sequences (Wikipedia: Sequene Alignment).(gus) But it is also used in plagiarism detection for both text documents and program code, as (lyo) argue.
(Olsen, 2009) lists a number of advantages to using sequence alignment algorithms as a generalised technique to identify small regions of similarity, ignoring large expansions of difference:

- Respects text order of documents.
- Does not require pre-identified blocks for comparison
- Can align similar passages directly, not as a region or block
- Not confused by extraneous similarities, like document topic.
- Spans variations in similar passages reflecting insertions and deletions.

- Core functions are language independent.

2.3.1.2 N-gram Measures

There are several approaches to employing the traits of a n-gram representation, but here is discussed only the approach taken by (Alberto Barron-Cedeno, 2009).

2.3.1.3 Lexical Structural Approaches

With this name are labelled the approches that consider structural similarities between texts, while also checking for the occurrence of common words between them. Here only the graph-edit-distance approach is presented

2.3.1.4 Edit Distance for Dependency Graphs

Syntactic and structural similarity of words. How similar are two words with respect to their syntactic function. Words that have similar syntactic roles, e.g. all personal pronouns, all verbs etc. Quantitative measure of the exact syntactic similarity between two words.
The idea here is that similar texts may preserve syntactic similarity while exchanging only content words. Structural similarity between texts can be computed, for example, by comparing sets for stopword n-grams(Stamatatos, 2011).
Assignment matrix $\rightarrow$ Assignment problem.

2.3.1.5 Semantic Relatedness Approaches

Semantic similarity or semantic relatedness is a concept whereby a set of documents or terms within term lists are assigned a metric based on the likeness of their meaning or semantic content (Wikipedia Semantic Relatedness) Concretely, this can be achieved for instance by defining a topological similarity, by using ontologies to define a distance between words (a naive metric for terms arranged as nodes in a directed acyclic graph, like a hierarchy, would be the minimal distance in separating edges between the two term nodes), or using means such as a vector space model to correlate words and textual contexts from a suitable text corpus. The concept of semantic similarity is more specific than semantic relatedness. However, much of the literature uses these terms interchangeably, along with terms like semantic distance.
In essence, semantic similarity, semantic distance, and semantic relatedness all mean, ”How much does term A have to do with term B?” The answer to this question is usually a number between -1 and 1, or between 0 and 1, where 1 signifies extremely high similarity/relatedness, and 0 signifies little-to-none.
Semantic relatedness determines how word senses are related to each other. These relations can be the: synonymy, antonymy, hyponymy, hypernymy or meronymy. The most computationally developed of these is synonymy, which for computational purposes is made equivalent to the less accurate metric of word similarity or semantic distance () (Martin og Jurafsky, 2011). Semantic similarity is a concept that tries to answer the question: are these two phrases saying the same thing? Thus, it asks for an appropriate representation of the meaning of a word.

2.3.1.6 Representing meaning

The field of computational semantics is concerned with the question of representing meaning in computationally friendly representations. One central concept is that of word meaning. But of course it alone is not sufficient for practical purposes. So the notion of greater structures of text are also of research concern.

Computational approaches to deriving word meaning.

1.Dictionary approaches. When one could use the definition of the dictionary 2.Distributional

Word-sense disambiguation is the process of defining the meaning of a word in a particular context.

2.3.1.7 Meaning of a sentence

An application should take into account all of these relations, in order to determine the semantic relation between to sentences. It can be useful for example, when two sentences have basically the same structure but are talking about two very different things. One classical approach has been Frege's principle of compositionality : Does the meaning of a sentence flow from using the rules of combining the words that constitute it?

2.3.1.8 Calculating Semantic Similarity

There are several approaches to calculating semantic relatedness. Semantic relationships between words

Topological Similarity

- Node Based
- Edge Based
- Pairwise
- Groupwise

Statistical Similarity

- Latent Semantic Analysis
- Explicit Semantic Analysis

2.3.1.9 Resnik Similarity

(Resnik, 1999)

Is a measure of semantic similarity in a IS-A taxonomy based on the notion of shared information content. It is based on the idea of evaluating semantic relatedness using network representations. A natural way to evaluate semantic similarity in a taxonomy is to evaluate the distance between the nodes corresponding to the items being compared-the shorter the path from one node to another, the more similar they are. Given multiple paths, one takes the length of the shortest one (Lee et al., 1993). Resnik argues about the problems with edge counting method:Edge counting method is sensitive to varying link distances. In addition,by combining a taxonomic structure with empirical probability estimates, it provides a way of adapting a static knowledge structure to multiple contexts. Let C be the set of concepts in an is-a taxonomy, permitting multiple inheritance. Intuitively, one key to the similarity of two concepts is the extent to which they share information in common, indicated in an is-a taxonomy. Below is shown a piece taken from the WordNet semantic resource.

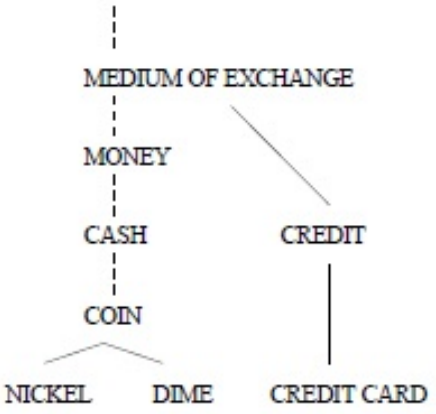

FIGURE 2.5: Fragment of the WordNet taxonomy. Solid lines represent is-a links; dashed lines indicate that some intervening nodes were omitted to save space.

The Resnik similarity is defined by this equation:

$$sim(c1, c2) = \max_{c \in S(c1,c2)} [-\log p(c)] \tag{2.5}$$

Where S(c1, c2) is the set of concepts that subsume both c1 and c2. Notice that although similarity is computed by considering all upper bounds for the two concepts, the information measure has the effect of identifying minimal upper bounds, since no class is less informative than its superordinates. For example, in Figure 1, coin, cash, etc. are all members of S(nickel, dime), but the concept that is structurally the minimal upper bound, coin, will also be the most informative. This can make a difference in cases of multiple inheritance; In practice one often needs to measure word similarity, rather than concept similarity. Using s(w) to represent the set of concepts in the taxonomy that are senses of word w, define:

2.4 Performance Evaluation -PAN Evaluation

PAN[4] is an evaluation lab for plagiarism detection systems that is held every year as part of the CLEF[5] conference. The PAN challenge offers a corpus of documents for both training and testing the approach. The corpus consists of a set of suspicious documents, hereafter referred as D_{susp}, a set of source documents, D_{src}, and a sets of XML document-pairs listing the suspicious and source documents that are related to each other, with pointers to the plagiarised passages. As stated in (Martin Potthast, 2012) the majority of the systems that were delivered to be tested on the task, implement three basic steps which are shown in the 2.6, taken from (Benno Stein, 2007). These steps are seen as standard for all the detectors.

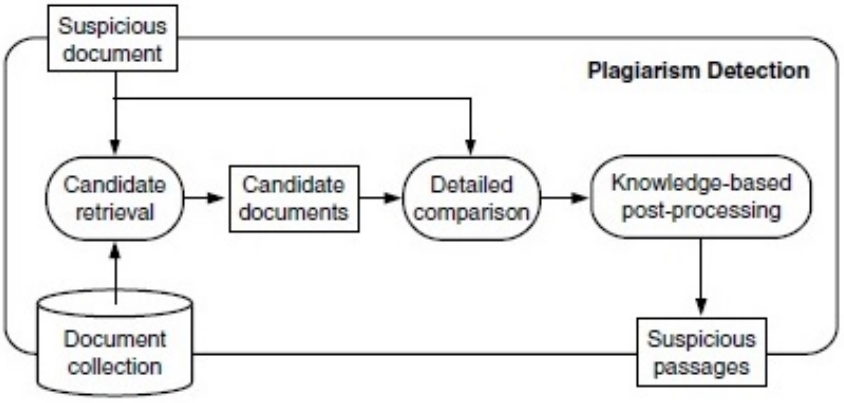

FIGURE 2.6: Generic retrieaval process to detect plagiarism

These steps consist in:

1. Candidate retrieval, which is the process of selecting a set of documents from the source documents corpus that most likely correspond to the source documents from which the suspicious documents have been derived.

2. The detailed comparison phase where each retrieved source document is compared to the plagiarised document, and passages of high similarity are extracted.

3. In the last phase the results go through a kind of post-processing where they are cleaned and filtered.

Before, the detectors were evaluated as a whole in their performance, but since many contestants were dropping the candidate retrieval step, given the relatively small amount of source documents available, the evaluation was changed to a stepwise one, where the candidate retrieval and the detailed analysis are judged separately. Here the focus shall be only on the measures used for the detailed analysis stage.
First, it is important to give a summary description of the text corpus which is used as

[4]http://www.pan.webis.de/
[5]Conference and Labs of the Evaluation Forum

input in training and in a possible cross-validation. This corpus contains documents grouped into two categories: suspicious-documents and source documents, where the suspicious documents are derived by obfuscating the source ones by the methods described in Section 2.1.3. It does also contain sets of XML documents, similar to those that a detection system conforming to the PAN criteria should produce as a result, grouped by the kind of plagiarism that they point to.
The plagiarism disclosed can be one of these cases:

- No-plagiarism. These documents point exactly to the passages that have nothing to do with each other.
- No-obfuscation. These are the passages that are copied verbatim.
- Random obfuscation. These passages are obfuscated by using artificial plagiarism and are of the kind of the text snippet shown in 2.1.3.
- Translation obfuscation. These are passages that are obfuscated by the method of back translation.
- Summary obfuscation. These are the ones that are paraphrased.

In 2012(Martin Potthast, 2012), the organizers of PAN changed their submission requirements from that of delivering submission results, to delivering the detection software itself. This was justified by the gains in the overall and continuous evaluation, as by getting hold of the system one could check the runtime, could give as input real plagiarism cases and it could be exposed to evaluation of later data sets,as for example that of PAN13. This last thing allows for a more fair comparison between the the systems that were originally tested on different data sets. Since the participants were allowed to deliver systems based on whatever programming language, the The TIRA (tir) platform was used for dealing with the complexity of the organizational issues. TIRA was also used in the training phase, as it returned the performance value for the submitted results after the system was run on the corpus described above.
Granularity measure:

$$gran(S, R) = \frac{1}{S_{\mathrm{r}}} \sum_{s \in S_{\mathrm{r}}} \mid R_{\mathrm{s}} \mid \tag{2.6}$$

The plagdet measure combines the aforementioned scores, together with the F-measure, denoted below, in order to provide a unique ordering of the participant's results.

$$plagdet(S, R) = \frac{F_1}{log_2(1 + gran(S, R))} \tag{2.7}$$

F-measure:

$$F_1 = \frac{2}{\frac{1}{r} + \frac{1}{p}} \tag{2.8}$$

The pros and cons about the *plagdet* score and a more thorough evaluation of this measure are to be found in (Martin Potthast, 2012).

Chapter 3

Related Work

The previous chapter presented some of the theoretical foundations of natural language processing that are to be found as initial or intermediate steps in all NLP applications. Moreover, it listed some algorithms that addressed the questions of structural -Graph Edit distance - or semantic - Resnik (1995), Lin (1998), LSA - similarity. But we have not yet discussed systems that employ specific strategies to complete the whole task of plagiarism detection on document level. Now we shall present some of the applied approaches to plagiarism detection which are related to our concern. We shall describe two full fledged detection systems and an overview of the solutions provided by the PAN12 participants. In the end we describe a system originally designed to handle tasks from digital humanities. These tasks are not directing the problem of plagiarism specifically but since they aim to find a multitude of relationships between texts, a solution of detecting plagiarism comes as a technical consequence.

3.1 Plagiarism Detection based on Graph Edit Distance

In Chapter 2. we mentioned the graph-based representation of natural language sentences. Here we discuss the plagiarism detection system described in (Røkenes, 2012) whose detection algorithm is based on this representation. The system implements functionality for reading files and processing them by NLP techniques; does some document retrieval on the basis of the post-processing format, a detailed analysis and outputs results of the analysis.
The data set used is that provided by PAN (experiments run on PAN10, PAN11), as described in Chapter 2. As it is based on the graph representation and dependency parsing whose scope is the sentence, the comparison algorithm should return a similarity value that denotes the similarity distance between two sentences. So, the first step in processing, is partitioning the documents into sentences. These sentences are then sent further to a processing pipeline that consists of these steps:

1. POS-tagging
2. Lemmatising
3. Dependency Parsing

The model used in POS-tagging is the *english-left3words-distim.tagger*, one of the models of the Stanford Part-Of-Speech Tagger[1].While the dependency graphs are created by the MaltParser[2], which is data-driven. The parsed sentences are then put in the database; the database used is MongoDB[3]. After these steps, each sentence in the database exists as a list of lemmatised tokens. To each token there is assigned a POS-tag, and an id, integer value, that denotes which token in the sentence it is dependent on. In the case when it is not dependent on any, this id is set to 0.
So when talking about "a sentence" hereafter, we intend this structure, shown below. This sentence is taken from the PAN13 set:

```
"id": ObjectId( "51ec1e1ee4b029dcab101afc" ),
"id": "suspicious-document00006.txt-135",
"filename" : "suspicious-document00006.txt",
"sentenceNumber": 135, "offset": 14156,
"length": 112,
"tokens": [  "id": "1", "word": "In", "lemma" : "in", "pos": "IN",
"rel": "8", "deprel" : "prep" ,  "id": "2", "word": "addition",
"lemma": "addition",
"pos": "NN", "rel": "1", "deprel" : "pobj"]
```

The main idea of the detection strategy goes like this: Let GED decide the similarity between each suspicious sentence and each source sentence and use this information to report for each suspicious document, which passages where plagiarised from which passages of one or more source documents.
As is also argued in (Martin Potthast, 2012) doing an exhaustive all-to-all comparison, does increase the recall of the result but the strategy is far from optimal given a set of source documents as large as the web for instance. In order to reduce the amount of sentences that each sentence of the suspicious documents is going to get compared to, the GED system employs a candidate retrieval phase, during which an amount of source sentences are discarded as not relevant for the analysis carried by the algorithm. After this retrieval step, the database elements have this form:

```
"source_file: source_documentxxxxx.txt, suspicious_sentence: xxx,
 candret_score: x.x, source_sentence: xxx,
 suscpicious_file :"suscpicious_documentxxxxx"
```

This is the input handed over to the "Detailed Analysis" phase whose main processing steps are pictured below:
As the internal workings of the program are described in (Røkenes, 2012) ,we shall not delve into many details here. The important thing to mention is that the distance calculation between these graph-sentences, is reduced to an assignment problem for a cost matrix, as described in Chapter 2. The cost matrix itself is a sentence-sentence matrix, where the cells represent the cost of substituting, deleting or inserting a node in a sentence.

[1] http://nlp.standford.edu/software/tagger.html
[2] http://maltparser.org
[3] http://www.mongodb.org/

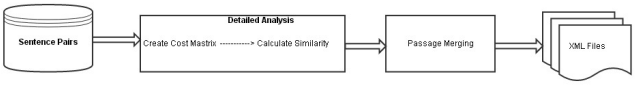

FIGURE 3.1: Process view of the ”Detailed Analysis”

3.1.1 Performance

As (Røkenes, 2012) describes in the resutls analysis, the system was tested on the PAN10 and PAN11 datasets. The *plagdet* score described in Chapter 2 is used to measure the performance. The system performes well enough to have been ranged 4th among the PAN11 participants, when only precision and recall scores are considered.

3.1.2 Further work

The system exhibits two main problems:

1. It does often not detect the sentences that lie near an already judged-as-plagiarised sentence as part of the plagiarised passage, even though they in reality are. (Røkenes, 2012) argues that the approach taken by the ”Passage Merging” package, by gluing together all sentences that are within a given distance, called *mergedist*, in order to reduce the *granularity* which is hold as an important trait by the PAN challenge, does not solve this problem. These sentences could in the first place, have been left out of the ”Detailed Analysis” phase by the ”Candidate Retrieval” phase. (Røkenes, 2012) thus proposes, to lower the grade of similarity required to classify sentences as plagiarised, for the sentences in the original suspicious douments (not the truncated ones) that lie after a detected sentence. The problem could be however solved by increasing the recall of the ”Candidate Retrieval” case. We also discuss another solution in Chapter 4.

2. Does not account for the semantics of the word tokens. The example below shows how the algorithm gives the same similarity match for the verb-pairs: *feed* vs *nourish* and *keep* vs *abandon.* This problem could be solved by integrating a semantic similarity measurement, as we again discuss in Chapter 4.

Mary fed the cat. Mary nourished the cat. GED for the two graphs: 0.25. Normalised: 0.0625 Edit-path:(feed $\rightarrow$ nourish) = 0.25
Mary kept the cat. Mary abandoned the cat GED for the two graphs: 0.25. Normalised: 0.0625 Edit-path:(keep $\rightarrow$ abandon) = 0.25

3.2 DKPro Similarity

DKPro Similarity[4] is an open source framework that contains a variety of text similarity measures (Daniel Bär, 2012). DKPro Similarity comprises a wide variety of measures, from ones based on simple n-grams and common subsequences to high-dimensional vector comparisons and structural, stylistic, and phonetic measures.Daniel Bär (2012) It was developed to work as a complement to the DKPro Core system which is in turn based on the Apache UIMA framework (David Ferrucci, 2004), but it is designed to be used also in a standalone mode. This thesis would be concerned with the latter, but in order to understand how DKPro Similarity can be part of another system, it is reasonable to present how DKPro Similarity and UIMA (Unstructured Information Management Architecture)[5] are coupled, as described in Daniel Bär (2012). The authors call this the "UIMA-coupled mode" as DKPro adheres to the UIMA-based language processing pipeline. DKPro Similarity does not support basic language processing techniques,i.e lemmatisation, POS-tagging etc, but it allows for using the components as part of existing experimental setups.

The picture below shows an architectural overview of this integration.

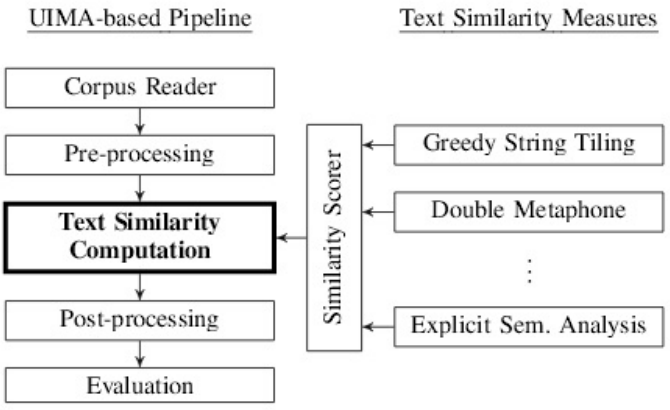

FIGURE 3.2: DKPro Similarity allows to integrate any text similarity measure(right) which conforms to standardized interfaces into a UIMA-based language processing pipeline (left) by means of a dedicated *Similarity Scorer* component (middle).

As showed in the figure, in this mode the text similarity computations of DKPro can be integrated directly in the UIMA pipeline. This does it easier to combine the DKPro functionalities with that of UIMA. As the authors argue, one could for example run a classification algorithm to some already generated UIMA similarity. The text similarity measures on the right can be easily exchanged since the system invokes them through a standardised interface, like in the code snippet below:

```
{ TextSimilarityMeasure m = new ResnikComparator();
  double similarity = m.getSimilarity(text1,text2);
```

[4]http://code.google.com/p/dkpro-similarity-asl/
[5]http://en.wikipedia.org/wiki/UIMA

The UIMA pipeline conforms to a standardized pipeline for text processing, as is also mentioned in the section above and in Chapter 2. That of reading a corpus, preprocessing (tokenization, POS-tagging,lemmatization, stopword filtering) and then the similarity calculation step.

3.3 Other

3.3.1 Semantic Textual Similarity

Semantic Textual Similarity(Eneko Agirre, 2012) (STS) measures the degree of semantic equivalence between two sentences. STS is related to both Textual Entailment (TE) and Paraphrase (PARA). STS differs from TE in as much as it assumes symmetric graded equivalence between the pair of textual snippets. In the case of TE the equivalence is directional, e.g. a car is a vehicle, but a vehicle is not necessarily a car. Additionally STS differs from both TE and PARA in that, rather than being a binary yes/no decision STS incorporates the notion of graded semantic similarity. STS provides a unified framework that allows for an extrinsic evaluation of multiple semantic components that otherwise have tended to be evaluated independently and without broad characterization of their impact on NLP applications. Such components include: word sense disambiguation and induction, lexical substitution, semantic role labeling, multiword expression detection and handling, anaphora and coreference resolution, time and date resolution, named-entity handling, underspecification, hedging, semantic scoping and discourse analysis.

example, for the measures that rely on path lengths (lch, wup, path) the tracing shows all the paths found between the concepts. Tracing for the information content measures (res, lin, jcn) includes both the paths between concepts as well as the least common subsumer. Tracing for the hso measure shows the actual paths found through WordNet,while the tracing for lesk shows the gloss overlaps in WordNet found for the two concepts and their nearby relatives. The vector tracing shows the word vectors that are used to create the gloss vector of a concept. We have incorporated WordNet Similarity into a generalized approach to word sense disambiguation that is based on semantic relatedness (Patwardhan, Banerjee, Pedersen 2003). This is implemented in the SenseRelate package. The premise of this algorithm is that the sense of a word can be determined by

3.3.2 PAIR and PhiloLine

PAIR[6] is a project of digital humanities with the goal of discovering similar passages in different texts. Here plagiarism is understood as only one of the many ways in which texts can be in relationship with each other. As the authors claim, the links that can exist between different texts are complicated and multifaceted, starting from pure stolen quotations to the more vague allusions and traces of influence. The aim of PAIR is to discover as many of these relations as possible, subsuming thus the highlighting of similar passages which can be cases of plagiarism. The implementation is based on a

[6]http://code.google.com/p/text-pair/

sequence alignment algorithm which is designed to identify similar passages in large corpora. There are two distinct streams of PAIR:

- PhiloLine: designed to perform all-against-all comparisons between documents loaded ina a PhiloLogic database. An entire corpus is indexed and compared against itself or another database to find text reuse. PhiloLine (PhiloLogic Alignment), a batch mode many-to-many aligner that compares all documents or document parts to all other documents or parts in the same database or between two collections. PhiloLine generates output either as static alignment reports or as structured data for subsequent search and analysis and is lightly dependent on PhiloLogic, our primary text analysis system.
- Text::PAIR : without specific bindings to PhiloLogic, supporting one-against-many comparisons. A corpus is indexed and incoming texts are compared against the entire corpus for text reuse.

Main goal of the project was sourcing l'encyclopedie : *"It is our expectation that systematic identification of the sources of the Encyclopédie will shed considerable light on the relationship of Enlightenment thought to French intellectual traditions, and also to currents of thought from classical antiquity to contemporary Western thinking."*

PAIR works by treating documents as ordered sets of n-grams or "shingles" formed by each overlapping sequence of n words in the document. Preprocessing, such as the removal of function words and short words and the reduction of orthographic variants is performed during shingle generation. This has the effect of folding numerous shingles into one underlying form for matching purposes, thus eliminating minor textual variations, which makes matching more flexible or "fuzzy." It also somewhat reduces the overall number of unique shingles, which aids speed of search.

Here are some important conclusions from the project:

- Adjusting the match parameters does that the matching sequences can have a degree a variation. This variability is required to identify possible borrowings which have significant errors, insertions, deletions, or other modifications.
- Detecting the full borrowing: Relaxing the maximum shingle gap to 8 or more allows for the identification of the full borrowing.
- how to solve the problem between detecting strict matches and matches that span between gaps : The available parameter adjustments for PhiloLine, and to a lesser degree for PAIR, are designed to balance the overall number of matches detected against the number of matches that a user would consider similar enough and salient enough to be of interest. If the matching is set too loosely, the user will have to wade through a large collection of short matches or common, stock phrases. If the settings are too strict, however, interesting matches that are short or heavily reworked may be missed. Speed and memory-use constraints also play a role in tuning the parameters for an optimal run. For PhiloLine, there are two sets of parameters to be adjusted. The first is for n-gram or shingle generation and the second for the matching process itself.

- While, as this experiment suggests, our approach is generally quite productive, success in sequence alignment is dependent upon using appropriate settings for the size of the passages to be aligned.
- One may set parameters to permit matches on small passages, but this increases the likelihood of aligning unrelated passages or generating too many trivial alignments.

- PAIR is a system which builds a database of shingles to be aligned against an unknown document, submitted by users
- Using VSM to detect similar docs. In a previous work (Allen, et al.), they examined the relationship of the Encyclopédie to a single contemporary reference work, the Jesuit Dictionnaire universel françois et latin (colloquially known as the Dictionnaire de Trévoux). It was widely assumed during the 18th century that the philosophes made extensive use of the Dictionnaire de Trévoux in the compilation of their work. Indeed, Jesuit critics of the Encyclopédie complained loudly of the extent to which entries were copied from earlier works, although among the possible sources of plagiarism the Trévoux dictionary was never explicitly mentioned. In order to attempt to detect possible borrowings, we used a general document similarity measure—the Vector Space Model (VSM)—to identify articles in the Encyclopédie which may have been borrowed, in whole or in part, from the Trévoux. The work used the text mining and machine learning extensions to PhiloLogic called PhiloMine

Chapter 4

Implementation

4.1 Contribution to the Graph Edit Distance System

The original idea of the contribution in form of implementing code, is that stated in Chapter 1. That formulation will not be changed since it can serve as a basis for further work, after the work done in this thesis.
The PAN organizers, had changed this years "Text Alignment" task such that an implementation of the Candidate Retrieval phase could be omitted, given that one could integrate in the system, the pairs.txt file handed out with the rest of the training corpus. This text file holds a list of suspicious-document - source-document pairs which tells which suspicious documents shall be compared with which source documents. The little script provided as a zip file accomplishes this task by reading pairs line for line, splitting each line in two variables *sus* and *src* and gathering all *src* that belong to a *susp* in one vector called *sources* such that for every unique *susp* there is a coresponding vector of source documents. When the program is run with readpairs.bash then it traverses all the 5 steps for each document, before it proceeds with the next.
By this approach, at least we are sure that the suspicious sentences are only going to get compared with sentences that are real candidates. The task of detecting the passages exactly is then handed over to the Detailed analysis step, which is unchanged.

4.1.1 Using DKPro

As mentioned in Chapter 3., DKPro in its standalone mode, offers a repository of text similarity algorithms, that can be integrated in an existing project.
If DKPro should be integrated with the system of (Røkenes, 2012), the LSR (Lexical Semantic Resources).The reason for choosing the Lexical Semantic Resource package is that of wanting to compare the semantic contents of the sentences. The module offering the LSA (Latent Semantic Analysis) would maybe work as well but there are two reasons for not choosing it: it would require much more recoding of the base system, and it is better suited to analyse the semantic content of whole documents, while PAN, as discussed in Chapter 2, is using artificially obfuscated documents, that in fact have no meaning.

There are several packages inside the LSR package: *aggregate*, *gloss* and*path*. The preferred package is that of *path* which contains the Resnik similarity algorithm (ResnikComparator.java) and the Lin similarity algorithm (LinComparator.java) among others.

Chapter 5

Evaluation

Every scientific work is justified by the results of its evaluation. The nature of this evaluation depends on the project itself and different disciplines impose different evaluation methods. Computer science is mostly concerned with delivering technological products that are usable in pragmatic contexts. Therefore it is natural to think of the evaluation of an AI project, in terms of the performance of the product that it has created. This performance could be judged by the professionals in a qualitative analysis, or we could measure it by how much it is widespread among the targeted users. This chapter employs the evaluation method proposed by the *"How Evaluation Guides AI Research"* (Paul R. Cohen, 1988) paper. The stages proposed for an evaluation process are first presented, then a short overview of how an evaluation plan for this work could have been set up, follows.

5.1 How does AI research proceed

The Paul R. Cohen (1988) presents a framework for doing AI research.The authors advocate from the start the importance of evaluation not only as a performance measure but also as an answer to a series of questions: why we are doing the research, why the tasks we choose are illustrative, why our views are avant-guard. Moreover they emphasize the necessity of showing how these planned tasks are implemented by the resulting system, how this system works and whether there are possibilities of improvement or the system has reached its best performance. A thorough answering of these questions would illuminate the audience of prospective researchers not only about the workings of the system, but more importantly how research of a particular topic should proceed. Besides this, evaluation is seen as a basis for accumulation of knowledge as documenting not only the answers that the project delivers but also the new questions that are produced on the way, it gives rise to new paths or other research topics. On the other hand, if a work is not evaluated, it would be difficult to replicate its results. The authors base their proposed framework of evaluation, on the five stages of a research process: refining the topic to a task; refine the view into a specific method; implement the method; design experiments to test the implementation; run the experiments. This is of course a bit idealized, they argue, but it can still work as a reference for standard AI research. In the following the evaluation of each of these steps is described.

5.1.1 Refine a topic to a task

The first stage presented is to refine the topic at hand to a task and a view. The ask itself is what we want the computer to do while the view is an imprecise idea of how we expect accomplish the task. The questions listed below are supposed to guide this process. These questions and the ones belonging to the other steps, are formulated by the authors. We have merely tried to understand their importance to the research activity.

1. Is the task significant and why?
2. Is your research likely to meaningfully contribute to the problem? Is the task tractable?
3. Is the task representative of a class of tasks?
4. Have any interesting aspects been abstracted away or simplified, if this problem has been previously defined?
5. What are the subgoals of the research? What key research tasks will be or have been addressed and solved as part of the project?
6. How do you know when you have successfully demonstrated a solution to the task? Is the task one in which a solution can be demonstrated?

Asking for the significance of a task is asking if the completion of the work is at all meaningful. If the specific problem addressed has been defined before, this would make the task not worthy of accomplishment. After the task has been formulated one has to make sure that it is tractable and how the solution could be classified as successful or not. It should also be clear what aspects have been left out or made simpler.

5.1.2 Design the Method

The questions of the second phase ask for a researcher to get to know the field and the latest results concerning the task at hand. In the case of producing a system for the PAN challenge, one has for example to review the approaches taken by the participants of the previous year. It would be rare if a given task in AI wouldn't rely on other methods, so it is important to explicitly denote which methods are relying on. The assumptions made do also need a clarification as the falsity of one of them would be disastrous during the development. One should also count with not getting excellent results, but its also important to distinguish poor results from results that mirror a total failure.

1. How is the method an improvement over the existing technologies?
2. Does a recognized metric exist for evaluating the performance of your method?
3. Does it rely on other methods?

4. What are the underlying assumptions?
5. What is the scope of the method?
6. When it cannot provide a good solution, does it do nothing or does it provide a bad solution?
7. How well is the method understood?
8. What is the relationship between the problem and the method?

5.1.3 Build a Program

1. How demonstrative is the program?
2. Is it specially tuned for a particular example?
3. How well does the program implement the method?
4. Is the program's performance predictable?

5.1.4 Design Experiments

This phase could be totally omitted if we were going to evaluate by the PAN measures, as the experiments are alredy set up.

1. How many examples can be demonstrated?
2. Should the program's performance be compared to a standard such as another program, or experts and novices, or its own tuned performance?
3. What are the criteria for good performance? Who defines the criteria?
4. Does the program purport to be general?
5. Is a series of related programs being evaluated?

5.1.5 Analyze the Experiment's Results

This stage is what most people regard as evaluation, thats why the previous stages and the questions they come with are often left out or not even considered as part of the evaluation. In addition to presenting the performance by the measures of evaluation, like for example using the *plagdet* value, one could oversee other important factors like the resources used to achieve those results.

1. How did the program performance compare to its selected standard?
2. Is the program's performance different from predictions of how the method should perform?
3. How efficient is the program in terms of space and knowledge requirements?
4. Did you learn what you wanted from the program and experiments?
5. Is it easy for the intended users to understand?
6. Can you define the program's performance limitations?
7. Do you understand why the program words or doesn't work?

5.2 Evaluation

As the implementation that this thesis provides is almost inexistent, there is not much to evaluate in terms of products or performance. The script talked about in Chapter 4 does actually integrate the pairs.txt file that would make up for the deficiencies stemming from the Candidate Retrieval phase. It's running time though its quite long, and it doesn't account for other exceptions originating from the system, that can show up during the running.

In terms of doing research in the field of AI, this work has been cause to two conclusions:

1. A hands-on approach is much more clarifying than reading theory whose answers one still doesn't have a question to. Even though no results were produced, trying to understand the coding of the Graph-edit-distance-based system and the approaches to change or contribute to it, were a basis for structuring the theoretical research. I guess that this research would have been much more thorough if coupled to an implementation problem at hand, as was the original plan.

2. When however, starting the work by exploring the theoretical foundations, it was found out that a top down approach was favourable. Which means, that reading about plagiarism detection systems and getting an understanding of their working in a high, use-case level created the necessity, and nonetheless the curiosity for the theoretical foundations that had given rise to these systems. The other problem with only reading theory without a specific task to be accomplished in mind, is that it never ends. It is actually one of the cases when more is more, because one paper asks to review another, and that maybe would lead one astray from a particular and somewhat narrower path.

Chapter 6

Conclusion and Further Work

6.1 Conclusion

The main goal of this thesis, was to explore ways of detecting plagiarised passages, by finding corresponding original passages from which they have drawn their form, content or both. Our first subgoal was to improve the performance of the GED system by attacking its two main shortcomings:

1. Making it recognize word replacement, i.e. detecting synonyms
2. Provide a smarter method for detecting plagiarism in sentences that lie near an already detected sentence

Neither of these goals were fully accomplished so they remain for a future work.

6.2 Further Work

Bibliography

Michael Jones. Back-translation: The latest form of plagiarism. *University of Wollongong*, 2009.

Håkon Drolsum Røkenes. Graph-based natural language processing:graph edit distance applied to the task of detecting plagiarism. 2012.

SPEECH and LANGUAGE PROCESSING An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Pearson, 2011.

Gobinda G. Chowdhury. Natural language processing. *University of Strathclyde, Glasgow G1 1XH, UK*, 2003.

A.M.Turing. Computing machinery and intelligence. *Mind , 59, 433-460 http://www.loebner.net/Prizef/TuringArticle.html*, 1950.

Alexandr Rosen Jan Hajic, Eva Hjicova. Formal representation of language structures. *Charles University, Prague, Czech Republic*, 2009.

Mira Seo. Plagiarism and poetic identity in martial. *http://muse.jhu.edu/journals/ajp/summary/v130/130.4.seo.html*, 2009.

Paul Clough. Old and new challenges in automatic plagiarism detection. *University of Sheffield*, 2003.

B.Martin. Plagiarism:a misplaced emphasis. *Journal of Information Ethics, Vol.3(2), 36-47*, 1994.

Martin Potthast. Technologies for reusing text from the web. *Faculty of the Media, Bauhaus-Universität, Weimar, Germany*, 2011.

James F. Allen. Natural language processing. 2003.

Ribeiro-Neto Baeza-Yates. Modern information retrieval. 2011.

Ruslan Mitkov Miranda Chong, Lucia Specia. Using natural language processing for automatic plagiarism detection. *4th International Plagiarism Conference;Northumbria University, Newcastle upon Tyne, UK.*, 2010.

Wang Shuai Du Cuixia-Wang Suhong Han Yong Kong Leilei, Qi Haolian. Approaches for candidate document retrieval and detailed comparison of plagiarism detection. *Notebook for PAN at CLEF*, 2012.

Lars Bungum Gleb Sizov-Björn Gambäck Andre Lynum Erwin Marsi, Hans Moen. Ntnu-core:combining strong features for semantic similarity. *Norwegian University of Science and Technology, Department of Computer and Information Science*, 2013.

Mona Diab Aitor Gonzalez-Agirre Eneko Agirre, Daniel Cer. A pilot on semantic textual similarity. *First Joint Conference on Lexical and Computational Linguistics*, 2012.

Mark Olsen. Sequence alignment and the discovery of intertextual relations. *ARTFL project*, 2009.

Paolo Rosso Alberto Barron-Cedeno. On automatic plagiarism detection based on n-grams comparison. *Natural Language Engineering Lab*, 2009.

Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language", volume 11, pages 95-130. 1999.

Matthias Hagen Jan Grassegger-Johannes Kiesel Maximilian Michel Arnd Oberländer Martin Tippmann Alberto Barron-Cedeno Parth Gupta Paolo Rosso Benno Stein Martin Potthast, Tim Gollub. Overview of the 4th international competition on plagiarism detection. *CLEF Evaluation Labs and Workshop*, 2012.

Martin Potthast Benno Stein, Sven Meyer zu Eißen. Strategies for retrieving plagiarized documents. *30th International ACM Conference on Research and Development in Information Retrieval (SIGIR 07)*, 2007.

Iryna Gurevych Daniel Bär, Torsten Zesch. Dkpro similarity: An open source framework for text similarity. 2012.

Adam Lally David Ferrucci. Uima:an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering,10(3-4):327-248*, 2004.

Adele E. Howe Paul R. Cohen. How evaluation guides ai research. *AI Magazine Volume 9 Number 4 AAAI*, 1988.